УДК 519.6:004.652.4

ПОДХОД К СОЗДАНИЮ БАЗ ДАННЫХ, ОСНОВАННЫЙ НА АЛГОРИТМАХ ГЕНЕРАЦИИ И ИДЕНТИФИКАЦИИ КОРТЕЖЕЙ

В.В. Кручинин, А.В. Титков, С.Л. Хомич

Томский университет систем управления и радиоэлектроники
E-mail: kru@ie.tusur.ru

Предложена оригинальная модель реляционной базы данных, в основе которой лежит представление доменов в виде деревьев И-ИЛИ. Разработаны оригинальные алгоритмы генерации и идентификации кортежей. Показана возможность существенного сжатия базы данных при небольших значениях мощностей доменов.

Введение

Анализ реальных информационных систем показывает, что зачастую домены таблицы имеют небольшие множества значений. Например, атрибуты: «сотрудник», «зарплата», «профессия», «возраст», «дата», «время» и т. д. [1]. Поэтому возможно предложить следующую идею: каждому кортежу декартового произведения множеств степенью n ставится в соответствие число и вместо кортежа в базе данных хранится это число. Для этого зададим отображение:

$$F : A_1 \times A_2 \times \ldots \times A_n \to N_n,$$

где $A_1 \times A_2 \times \ldots \times A_n$ – декартово произведение множеств; N_n – множество номеров $\overline{0,n}$.

Если F биективно, то можно задать обратное отображение:

$$F^{-1} : N_n \to A_1 \times A_2 \times \ldots \times A_n .$$

Таким образом, биективное отображение F задает алгоритм идентификации кортежа декартового произведения:

$$num = Rank(D, a),$$

где $a \in A_1 \times A_2 \times \ldots \times A_n$, $num \in N_n$, D – описание множеств декартового произведения $A_1 \times A_2 \times \ldots \times A_n$. А отображение F^{-1} задает алгоритм генерации значения кортежа по номеру:

$$a = Generate(D, num),$$

где $a \in A_1 \times A_2 \times \ldots \times A_n$, $num \in N_n$, D – описание множеств. Тогда отношение $R \subset A_1 \times A_2 \times \ldots \times A_n$, $num \in N_n$ можно однозначно представить подмножеством целых чисел $NUM \subset N_n$.

Используя алгоритмы Rank и Generate можно предложить следующую структуру базы данных (рис. 1). При записи кортежа в базу данных работает алгоритм Rank, который присваивает номер данному кортежу. Далее этот номер хранится в базе данных. При выборке данных из базы работает алгоритм Generate, который по заданному номеру получает кортеж. Важным элементом является описание множеств декартового произведения D. Рассмотрим подробнее способы организации D, Rank, Generate.

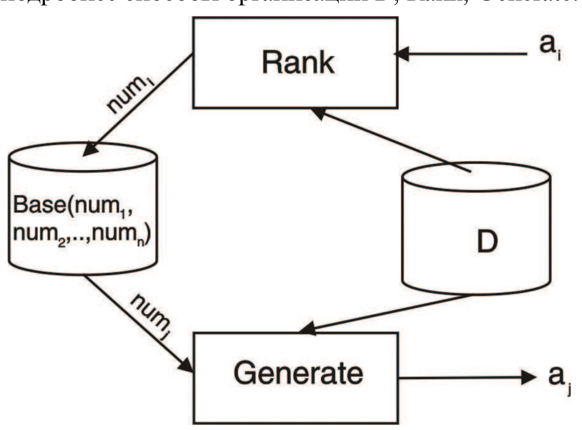


Рис. 1. *Описание структуры базы данных*

1. Алгоритмы генерации и идентификации на основе деревьев И-ИЛИ

Рассмотрим способ построения описаний множеств значений доменов D, алгоритмов идентификации *Rank* и генерации *Generate*. В качестве такого инструмента предлагается использовать деревья И-ИЛИ [2]. Правила построения дерева И-ИЛИ следующие:

1. Если некоторое множество разбивается на n множеств $\{A_i\}_{i=1}^{n}$, то это разбиение можно представить ИЛИ-узлом. При этом должно быть выполнено следующее условие:

$$\bigcap_{i=1}^{n} A_i = \varnothing. \quad (1)$$

2. Если искомое множество является комбинацией элементов из n множеств, то данное преобразование представляется И-узлом. В этом случае, условие (1) не требуется, необходимо, что бы комбинация была уникальной.

Листьями такого дерева являются элементы или множества, разбиение которых не производится. Используя два этих правила можно строить деревья И-ИЛИ для описания различных классов множеств.

Вариантом дерева И-ИЛИ назовем дерево, которое получается из заданного путем отсечения дуг кроме одной у всех ИЛИ-узлов. Корнем варианта будет являться корень дерева И-ИЛИ. На рис. 2 показан пример дерева И-ИЛИ и всех его вариантов.

Если дерево описывает некоторое множество, то вариант описывает один элемент множества. Тогда общее число вариантов в дереве (или мощность множества) можно вычислить по формуле:

$$\omega(z) = \begin{cases} \sum_{i=1}^{n} \omega(s_i^z) & \text{для ИЛИ-узла} \\ \prod_{i=1}^{n} \omega(s_i^z) & \text{для И-узла} \\ 1 & \end{cases}, \quad (2)$$

где z – рассматриваемый узел дерева; s_i^z – множество сыновей узла z; n – число сыновей.

Тогда, зная $\omega(z)$ для каждого узла, можно предложить следующий алгоритм генерации варианта (*Generate)*:

1. Корень дерева записывается в вариант и заносится в стек $Stack \overset{push}{\leftarrow} < s_{root}, L >$.
2. Из стека вынимается пара $< z, l_z > \overset{pull}{\leftarrow} Stack$. Если стек пуст, то завершить работу.
3. Определяется тип текущего узла. Если это И-узел, то переход на шаг 4, иначе переход на шаг 5.
4. Все сыновья $\{s_j^z\}_{j=1}^{m}$ рассматриваемого узла z записываются в данный вариант V, вычисляется $l_A(s_i^z)$, используя выражение

$$l_A(s_i^z) = \begin{cases} \dfrac{l_A(z)}{\prod_{j=1}^{i-1} \omega(s_j^z)} \bmod \omega(s_i^z) & i > 1 \\ l_A(z) \bmod \omega(s_i^z) & i = 1 \end{cases}, \quad (3)$$

и пары $<s_j^z, l_A(s_j^z)>$ заносятся в стек.

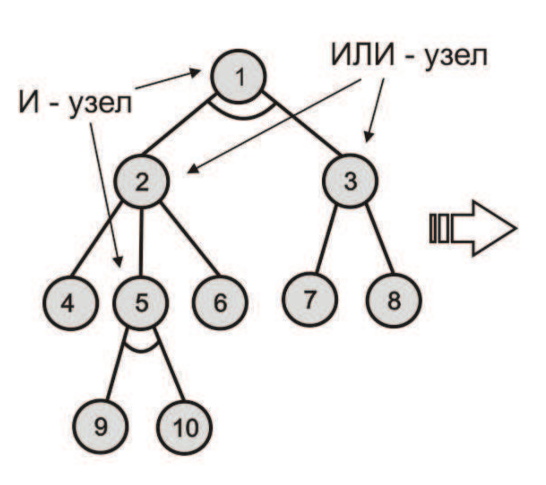
5. Если это ИЛИ-узел, то, используя выражение

$$l_o(s_k^z) = \begin{cases} l_o(z) & \text{при } l_o(z) < \omega(s_k^z),\ k = 1 \\ \min_k [l_o(z) - \sum_{j=1}^{k} \omega(s_j^z)] & \text{при } l_o(s_k^z) \geq 0,\ k > 1 \end{cases}, \quad (4)$$

определяется единственный сын s_k^z и $l_o(s_k^z)$. Сын записывается в вариант V, а пара $<s_k^z, l_o(s_k^z)>$ заносится в стек.

6. Переход на шаг 3.

Анализ данного алгоритма показывает, что временная сложность пропорциональна количеству узлов, которые заносятся в стек, следовательно, пропорциональна числу узлов в варианте. При этом количество делений равно числу сыновей всех И-узлов варианта плюс число сложений и сравнений для ИЛИ узлов (см. выражения 3 и 4).



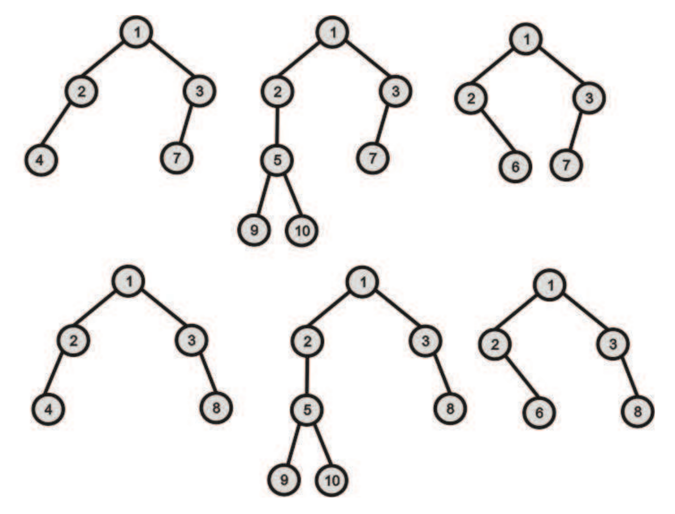


Рис. 2. Дерево И-ИЛИ и все его варианты

Теорема. Пусть дано дерево И-ИЛИ D и для каждого узла $z \in D$ имеется $\omega(z)$, тогда алгоритм генерации задает биективное отображение $G:N_n \to W$, где W – множество всех вариантов. Покажем что отображение G инъективно, т. е. для $\forall i \neq j$ следует, что $G(i) \neq G(j)$. Это утверждение основывается на рассмотрении выражений (3) и (4). Выражение (3) заданным числам i и j ставит в соответствие два разных набора чисел для сыновей узла И, поскольку происходит преобразование чисел i и j в числа со смешанными основаниями, представленными $\{\omega(s_i^z)\}_{i=1}^n$ для узла z. Выражение (4) числам i и j для узла ИЛИ, получает две разных пары (k, l), где k – номер узла, l – значение $\omega(s_i^z)$. Таким образом, алгоритм *Generate* задает инъективное отображение $G:N_n \to W$. Поскольку множества N_n и W конечны и мощности их равны, следовательно, отображение $G:N_n \to W$ биективно. Из этого следует, что $\forall V \in W \exists i \in N$, что $V=G(i)$, следует, что для любого варианта дерева И-ИЛИ можно найти единственный номер i. Построим алгоритм нумерации варианта для данного дерева И-ИЛИ. Для этого необходимо найти сопоставление варианта V в дереве D и нахождение соответствующего номера i.

Сопоставление производится следующим образом:

1. Первоначально в стек M_1 заносится корень варианта V, в стек M_2 корень дерева D.
2. Если стек M_1 пуст, то завершить работу алгоритма.
3. Из стека M_1 извлекается узел варианта dv и из стека M_2 извлекается узел d.
4. Если это узлы И, то все сыновья dv заносятся в стек M_1, а сыновья d заносятся в M_2. Переход на шаг 2.
5. Если это узлы ИЛИ, то сын dv ищется в множестве сыновей узла d. Если найдено совпадение, то сыновья заносятся в стек. Переход на шаг 2.
6. Если dv и d листья, то они удаляются из стека.

Вычисление номера начинаем производить с рассмотрения листьев варианта V. Все листья варианта имеют значения $\omega(z)=1$.

После того как сопоставление найдено, выполняем следующие действия:

1. Для каждого И-узла z вычисляем

$$l_z = l_1 + \omega(s_1)(l_2 + \omega(s_2)(...(l_n)\omega(s_{n-1}))...)),$$

где $\{s_i\}_{i=1}^n$ – сыновья узла z, а $\{l_i\}_{i=1}^n$ – соответствующие номера, полученные для сыновей.

2. Для каждого ИЛИ-узла вычисляем

$$l_z = \sum_{i=1}^{k-1} \omega(i) + l_1,$$

где k – номер соответствия для узла ИЛИ в дереве D, l_1 – номер варианта для этого сына. Рекурсивно производим вычисления номера, пока не достигнем корня дерева. Полученное число l_z для корня варианта будет номером варианта, т. е. $V=R(l_z)$. Очевидно, что $l_z \leq \omega(z)$. Таким образом, для множества, представленного деревом И-ИЛИ, можно создать алгоритмы *Rank* и *Generate*.

2. Преобразование таблицы атрибутов в дерево И-ИЛИ

Рассмотрим построение дерева И-ИЛИ для таблицы атрибутов. Поскольку значение $a \in A_1 \times A_2 \times ... \times A_n$ является комбинацией элементов из множеств $\{A_i\}_{i=1}^n$, то корень дерева будет И-узлом, имеющих n сыновей, каждый i-й сын соответствует множеству A_i, графическое изображение такого соответствия показано на рис. 3.

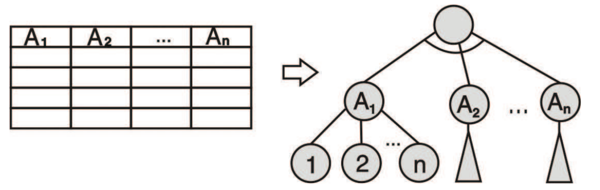


Рис. 3. *Соответствие между таблицей и деревом И-ИЛИ*

Общее число множества значений вычисляется по формуле:

$$\omega(T) = \prod_{i=1}^{n} \omega(A_i).$$

Далее для каждого множества A_i строится свое дерево И-ИЛИ. В общем случае можно представить множество значений A_i

1. справочником;
2. числовым интервалом;
3. деревом И-ИЛИ.

Для представления множества уникальных объектов, которые используются в базе данных некоторого домена, используется справочник. Справочник имеет две части, первая часть содержит пронумерованные уникальные объекты, вторая часть резервная, предназначена для внесения новых объектов. Соответствие между справочником деревом И-ИЛИ показано на рис. 4. Справочник представляется ИЛИ-узлом, а все сыновья являются элементами справочника. Тогда общее число вариантов дерева (или элементов множества) равно:

$$\omega(A_i) = n + m.$$

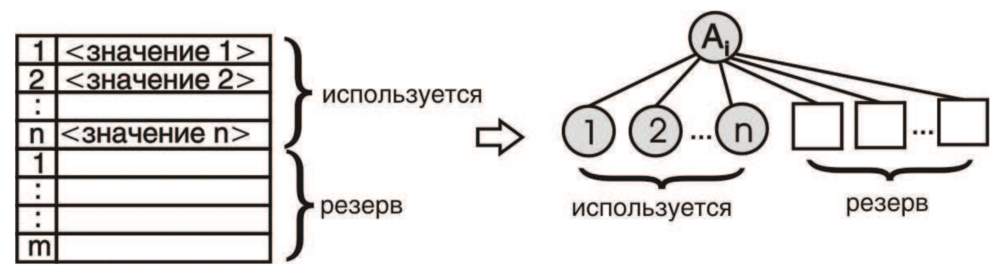


Рис. 4. *Соответствие между справочником и деревом И-ИЛИ*

Для представления числового интервала задается границы и шаг, тогда данное множество можно представить деревом И-ИЛИ, которое имеет ИЛИ-узел в качестве корня, а сыновья, конкретные значения чисел из этого интервала. Графическое изображение такого дерева показано на рис. 5.

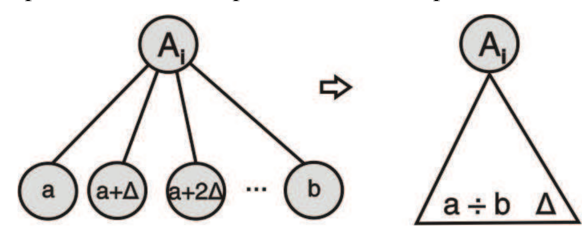


Рис. 5. Дерево для представления числа

Тогда общее число вариантов (элементов множества) будет:

$$\omega(A_i) = \frac{b-a}{\Delta}.$$

Множество значений A_i может быть представлено деревом И-ИЛИ. Рассмотрим несколько наиболее распространенных примеров. Если A_i это дата, то ее можно представить следующим деревом И-ИЛИ (рис. 6):

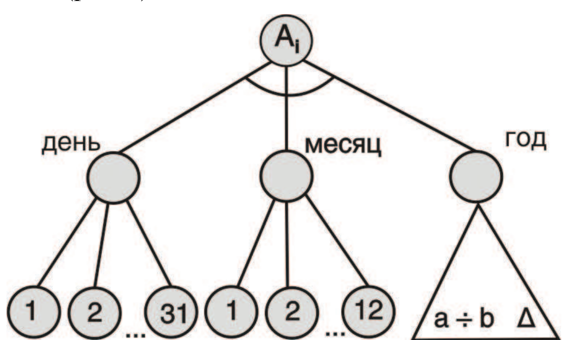


Рис. 6. Дерево И-ИЛИ для представления даты

Здесь при описании даты год представлен некоторым числовым интервалом. Например, 1950–2050, Δ=1. Тогда общее число вариантов может быть представлено формулой:

$$\omega(\textit{Дата}) = \omega(\textit{день}) \cdot \omega(\textit{месяц}) \cdot \omega(\textit{год}).$$

Аналогично может быть представлен атрибут «время». На рис. 7 показано дерево И-ИЛИ для представления атрибута.

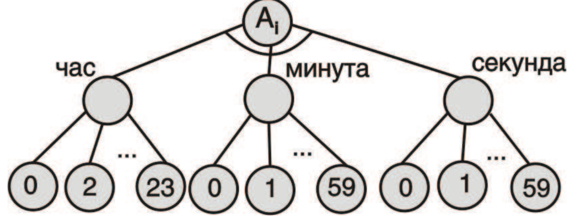


Рис. 7. Дерево И-ИЛИ для представления атрибута «время»

Тогда общее число вариантов может быть представлено формулой:

$$\omega(\textit{Время}) = \omega(\textit{час}) \cdot \omega(\textit{минута}) \cdot \omega(\textit{секунда}).$$

Таким же образом, можно представить атрибуты «Зарплата» и «Коэффициент» (рис. 8).

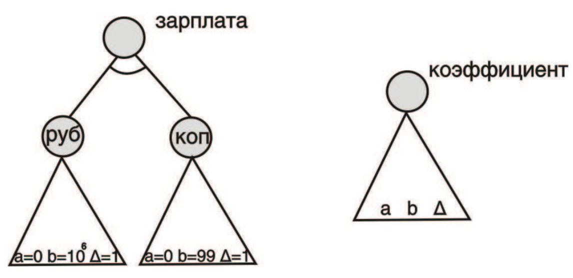


Рис. 8. Деревья И-ИЛИ для представления атрибутов «зарплата» и «коэффициент»

3. Оценка мощности множества вариантов дерева И-ИЛИ для представления декартового произведения

Пусть дано декартово произведение множеств $A_1 \times A_2 \times A_3 \times A_4 \times A_5 \times A_6 \times A_7 \times A_8$, где:

$A_1 \to$ФИО;

$A_2 \to$стаж;

$A_3 \to$оклад;

$A_4 \to$проработанное время;

$A_5 \to$коэффициент;

$A_6 \to$дата;

$A_7 \to$должность;

$A_8 \to$районный коэффициент.

Пусть в фирме работает 1000 чел., текучесть кадров 100 чел. в год. Тогда

$$\omega(A_1) = (1000 + 2000) = 3000 = 3 + 10^3$$

$$\omega(A_2) = [(0,100), \Delta = 1] = 100 = 10^2$$

$$\omega(A_3) = 100[(0,10^6), \Delta = 1] = 10^8 \to$$

$$\omega(A_4) = 24 \cdot 30 = 720 \approx 10^3$$

$$\omega(A_5) = [(0,1), \Delta = 0,001] = 100 = 10^2$$

$$\omega(A_6) = 74400 \approx 75000 = 75 \cdot 10^5$$

$$\omega(A_7) = (200 + 800) = 1000 = 10^3$$

$$\omega(A_8) = [(1,2), \Delta = 0,1] = 10 = 10^1$$

Тогда

$$\omega(D) = \prod_{i=1}^{8} \omega(A_i) = (3 \cdot 10^3) \cdot (10^2) \cdot (10^8) \times$$

$$\times (720) \cdot (10^2) \cdot (75 \cdot 10^3) \cdot (10^3) \cdot (10^1) = 1,62 \cdot 10^{27}.$$

Таким образом, все множество картежей меньше, чем $1,62 \cdot 10^{27}$ и $10^{28} < 2^{93}$.

Тогда, для представления номера картежа $\prod_{i=1}^{8} A_i$ необходимо 93 бита или 12 байт. Оценим теперь размер дерева И-ИЛИ D для описания множества картежей. Общий размер дерева вычисляется по формуле:

$$Size(D) = \sum_{i=1}^{8} Size(A_i),$$

где $Size(A_1) = n \cdot Size(\textit{ФИО}) = 1000 \cdot 60$.

Все множества, описываемые интервалами значений, имеют фиксированную длину Const.

$$Size(A_7) = n \cdot Size(должность) = 200 \cdot 40.$$

Тогда $Size(D)$=60000+8000+$Const$·6<70000. Это означает, что объем базы данных будет равен:

$$S_D = N \cdot 12 + Size(D),$$

где N – число кортежей в базе; $Size(D)$ – размер описания дерева И-ИЛИ для описания множеств.

Предположим, что в базе имеется 1000 кортежей, тогда:

$$S=12000+70000=82000 \text{ байт.}$$

Оценим размер таблицы при традиционном подходе:

$$S_T = N \cdot Size(Строка) = (1000) \cdot (128) = 128\,000 \text{ байт.}$$

Тогда коэффициент сжатия будет равен:

$$k = \frac{S_T}{S_D} = \frac{N \cdot Size(Строка)}{N \cdot Size(num) + Size(D)} = \frac{Size(Строка)}{Size(num) + \frac{Size(D)}{N}}.$$

В нашем примере этот коэффициент равен 128000/70000=1,82.

Однако для больших значений N можно получить значительный эффект, при условии, что размер D не пропорционален N. Тогда такое построение базы данных позволяет в разы сократить размер базы данных.

Заключение

Предложенный подход к построению баз данных, основанный на построении алгоритмов генерации и идентификации кортежей, позволяет существенно сжимать объемы хранимой информации. Особенно для тех баз данных, для которых домены имеют фиксированный размер. Однако реальные размеры справочников могут иметь размеры 10^6 и более. Тогда значения $\omega(D) = \prod_{i=1}^{n} \omega(A_i)$ могут превышать значение 10^{100}. Переход на архитектуру процессоров с разрядностью регистров 64 и 128 решит возникающие трудности по обработке больших целых чисел.

СПИСОК ЛИТЕРАТУРЫ

1. Рубанов В.В. Способы отображения объектов в реляционных базах данных // Труды ИСП РАН. – 2002. – Т. 3. – С. 137–162.
2. Кручинин В.В. Алгоритмы и перечислительные свойства деревьев И-ИЛИ // Вестник ТГУ. – 2004. – № 284. – С. 181–184.

УДК 621.397